An Approach to Cooperation: Planning and Communication in a Distributed Problem Solving Network

Edmund H. Durfee

COINS Technical Report 86-09
March 1986

Abstract

Cooperation between intelligent agents requires that the agents understand the consequences of their actions and interactions. Agents must plan to perform beneficial actions, and must communicate about their intentions to promote helpful interactions and to avoid harmful interactions. In this paper, I describe an approach for increasing coherent cooperation among agents that are working together on a single problem. Initially, a simple model of cooperation among self-interested agents is presented. Next, the methodologies for achieving cooperation between problem solving agents are investigated and the important connection between distributed problem solving and distributed planning is recognized. Subsequently, I discuss preliminary research into improving cooperation in distributed problem solving networks by introducing planning and communication of *meta-level* (coordination) information. Finally, further research is proposed to improve on the planning and communication mechanisms so that cooperation can be achieved in a variety of problem solving situations.

This research was sponsored, in part, by the National Science Foundation under Grant MCS-8306327, by the National Science Foundation under Support and Maintenance Grant DCR-8318776, by the National Science Foundation under CER Grant DCR-8500332, and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049-041.

An Approach to Cooperation: Planning and Communication in a Distributed Problem Solving Network

Edmund H. Durfee
Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

March 1986

Abstract

Cooperation between intelligent agents requires that the agents understand the consequences of their actions and interactions. Agents must plan to perform beneficial actions, and must communicate about their intentions to promote helpful interactions and to avoid harmful interactions. In this paper, I describe an approach for increasing coherent cooperation among agents that are working together on a single problem. Initially, a simple model of cooperation among self-interested agents is presented. Next, the methodologies for achieving cooperation between problem solving agents are investigated and the important connection between distributed problem solving and distributed planning is recognized. Subsequently, I discuss preliminary research into improving cooperation in distributed problem solving networks by introducing planning and communication of *meta-level* (coordination) information. Finally, further research is proposed to improve on the planning and communication mechanisms so that cooperation can be achieved in a variety of problem solving situations.

1. Introduction

There is no mystery to why people cooperate. Cooperation occurs when each person believes that he or she will benefit more by cooperating than by acting in some other way.

This research was sponsored, in part, by the National Science Foundation under Grant MCS–8306327, by the National Science Foundation under Support and Maintenance Grant DCR-8318776, by the National Science Foundation under CER Grant DCR-8500332, and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049–041.

Similarly, groups of people will cooperate for their mutual benefit: businesses cooperate to increase profits, and nations cooperate in part to improve security (increase the survival probability of their people). In such situations, the individual cooperating *agents*, be they people or groups of people, only cooperate to improve their own self-interests. The prevalence of cooperation in human society, and in the world in general, indicates that there are many ways that selfish agents can interact for their mutual benefit.

Despite the prevalence of cooperation, however, it is still a poorly understood phenomenon. The advent of computers as agents in human environments has led to many interesting problems that stress issues in cooperation. For example, cooperation among humans is facilitated because people have an understanding of each other—each can predict the other's actions because they have so much in common (their humanity). Human-computer interactions are often fraught with frustration and misunderstanding because neither agent has an adequate view of why the other is behaving as it is [45]. Furthermore, computer-computer interactions have been similarly problematic because computers have primitive, if any, abilities to understand other computers. Unlike natural systems where such understanding evolved with the species, artificial systems must explicitly be given such understanding.

The research proposed in this paper is a step toward this end. The artificial systems studied use artificial intelligence techniques to solve problems. A network of these problem solvers perform *distributed problem solving* by cooperating as a team to solve a single problem. The principal focus of this research is on developing mechanisms that allow each problem solver to understand what other problem solvers are doing. Since there are many ways that problem solvers can cooperate (partition the problem and work in parallel on complementary pieces, duplicate each other's work to corroborate results, develop diverse solutions using individual expertise to avoid overlooking possibilities, etc.), distributed problem solving provides a rich environment to study issues in cooperation among artificial systems.

To adequately address issues in cooperation among intelligent agents, it is necessary both to more fully define what these terms mean and to outline past research in this area. In the next section, I develop a simple view of cooperation among agents. Section 3 focuses on cooperation among intelligent agents—agents that can apply knowledge to decide what actions (cooperative or not) they will take. To cooperate effectively while problem solving (or performing any other task), intelligent agents must plan their actions and interactions; the relevant issues in distributed planning and problem solving are outlined in Section 3. Section 4 describes preliminary mechanisms that improve cooperation among agents. Each agent possesses a planner that increases the agent's understanding of its own activities. By exchanging this type of information, agents increase each other's understanding of overall network activity. Because these mechanisms were primitive and limited, the main focus of the research outlined in Section 5 will be the implementation of more sophisticated and versatile mechanisms. It is hoped that these mechanisms and the approach to improving cooperation that they represent will not only improve network problem solving in a distributed problem solving network, but will also act as a foundation for improving cooperation among artificial systems in general.

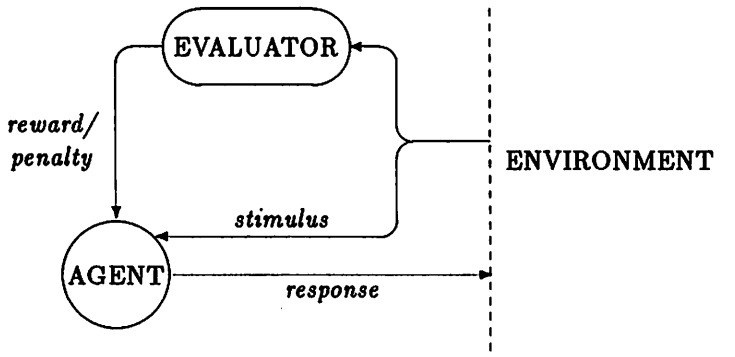


Figure 1: A Model of Agent/Environment Interaction.

2. Cooperating Agents

An agent is something that reacts to its environment. In our anthropocentric view of the world, we usually consider people as the ultimate agents. In many environments, however, the principal agents might be computers, cells, transistors or molecules. An agent is affected by its environment: an undesirable environment can destroy an agent (kill a person or cell, crash a computer, burn a transistor, break apart a molecule) while a desirable environment can increase an agent's chances of survival. By its responses to the environment, an agent may make the environment more or less desirable.

A model of agent and environment interaction is shown in Figure 1. At any given time, an agent is affected by the environment through some set of *stimulus* information. The stimulus is evaluated to determine the desirability of the environment for the agent, and the agent is given a *reward* or *penalty* based on this evaluation. The evaluation can occur within the agent (people can often evaluate their environments) or it may be done by a separate entity (for example, a computer often relies on a human to evaluate its performance). Finally, the agent can use the stimulus information and the reward/penalty to generate an appropriate *response*. The response modifies the environment, and the cycle repeats.

2.1 Types of Agents

The agent model is illustrated with two very different examples. First, consider a catalyst molecule that reacts in the presence of two compounds to generate a product. The reaction also causes heat to be released, and the catalyst becomes more reactive at high temperatures. Thus, the environment consists of the chemical mixture and its temperature, the agent is the catalyst, and the evaluator is the reactivity of the catalyst—the catalyst-agent is increasingly rewarded as the temperature rises. In the presence of the appropriate compounds the catalyst-agent responds by forming the product and heat, thereby increasing its reward.

As a second example, consider an office environment with the boss as the agent. The boss-agent evaluates the office environment in terms of its productivity and is rewarded

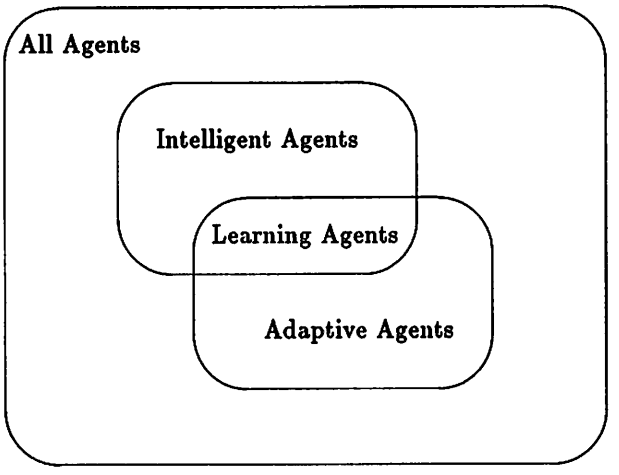


Figure 2: A Diagram of Agent Subclasses.

for high productivity. When productivity falls, the boss chooses a response to raise it again. For example, the boss may decide to maintain stricter discipline. If productivity continues to fall despite this response, the boss could change the response and instead promote company spirit.

Although the simple model captures the essential aspects of both the catalyst-agent and boss-agent scenarios, the different complexities of these agents affect our perception of them. If the catalyst-agent were more reactive at low temperatures, then its response to the environment would lower its reward; the catalyst-agent cannot alter or qualify this response. The boss-agent, on the other hand, can develop different responses to the same stimulus by applying its own knowledge to the situation and by monitoring the effects of previous responses. I therefore define important agent subclasses (Figure 2). An *intelligent* agent can apply knowledge to a situation to develop and rank possible responses; knowledge-based computer systems as well as higher forms of animal life can be classified as intelligent agents. An *adaptive* agent can relate responses to their effects on the environment (the rewards) and will modify the responses to increase its rewards; neurons and neuron-like elements are examples of adaptive agents. Agents that are both adaptive and intelligent can be classified as *learning* agents, the most common example of which are humans.

2.2 Multi-Agent Environments

If an agent is the only active force in an environment, it is responsible for all changes in that environment. Such an agent indeed controls its own destiny. In general, however, agents share their environments with other agents, and may therefore interact since the actions of each agent may affect the environment perceived by the others.

Biologists have developed terminology for interactions between agents [23]. If the actions of each agent increase the rewards of the other agents, then the agents are *cooperative*. For example, if a chemical reaction had numerous catalyst-agents like those described above, the actions of each would raise the reaction temperature and thus increase the re-

		AGENT-2		
		Reward	Penalize	No Effect
AGENT-1	Reward	*Cooperation*	*Exploitation*	*Beneficial Asymmetry*
	Penalize	*Exploitation*	*Competition*	*Detrimental Asymmetry*
	No Effect	*Beneficial Asymmetry*	*Detrimental Asymmetry*	*Co-existence*

Figure 3: Possible Interactions Between Agents.

wards for all. Agents that are rewarded for mutually exclusive environments, on the other hand, are *competitive*. In the chemical reaction, for example, if one catalyst-agent raises the temperature and prefers higher temperatures while another catalyst-agent lowers the temperature and prefers lower temperatures, then the actions of one penalize (lower the reward of) the other. Finally, agents are *co-existent* if each performs actions that do not affect the rewards of the others. For example, if one catalyst-agent is rewarded for high temperatures while another is rewarded for high acidity, then since temperature and acidity are independent aspects of the environment each agent can alter its own reward without affecting the reward of the other.

Alternative combinations of these interactions can occur as well (Figure 3). When one agent rewards the other while the other agent penalizes the first, the interaction is *exploitive*. For example, one catalyst-agent (cat_1) may be reactive at high temperatures and acts to increase acidity, while another catalyst-agent (cat_2) may be reactive at low acidity and acts to increase the temperature. When together, the actions of cat_2 reward cat_1 but cat_1 penalizes cat_2: cat_1 exploits cat_2.[1] When the actions of one agent affect the other but not *vice versa*, the interaction is *asymmetric*. The effect is beneficial if it rewards the agent and detrimental if it penalizes the agent. If a catalyst-agent is rewarded for high temperatures but a chemist immerses the reaction in cold water, then the catalyst-agent is penalized by the actions of the chemist-agent while the chemist-agent may be neither rewarded nor penalized. This is an example of detrimental asymmetry.

Agents involved in asymmetric interactions will view each other's activities differently. The unaffected agent, by definition, will not perceive how its actions affect the other agent; from its viewpoint, the other agent does not exist. Conversely, since the affected agent cannot influence the actions of the unaffected agent, it will perceive these actions not as the responses of an agent but as uncontrollable changes to the environment. Agents therefore recognize two types of environmental changes: actions and events. *Actions* are the perceived responses of agents that modify the environment. Through cooperative, compet-

[1] Of course, this cannot continue indefinitely. If heat and acidity do not dissipate, then eventually cat_2 will become entirely inactive and exploitation will cease. If heat and acidity do dissipate, equilibrium (possibly cyclic) can occur.

itive, or exploitive interactions with other agents, an agent can influence the actions in its environment. *Events* are changes to the environment that are beyond an agent's control because they are caused by agents that it cannot influence. For example, meteorologists around the country might exchange information and predictions to influence each other's view of the weather, but this activity will not affect the weather. A meteorologist will thus view the activities of other meteorologists as actions, while changes in the weather are events that he or she cannot control (and may even have difficulty predicting). Since agents have control over actions but not events, planning research (including that outlined later in this paper) often distinguishes between the two [60].

2.3 The Evolution of Cooperation

Agents are self-interested. An agent responds to its environment in an attempt to increase its own reward. Sometimes the response may decrease the agent's own reward; unintelligent or unadaptive agents may be unable to avoid such responses. Sometimes the response may increase or decrease the rewards of other agents and may in turn elicit certain responses from them. Intelligent agents can use their knowledge to intentionally affect other agents, while interactions between unintelligent agents are always unintentional.[2] Since both types of agents can cooperate, intent is not a prerequisite for cooperation. In this paper, however, the emphasis is on providing intelligent agents with the knowledge they need to cooperate intentionally for their mutual benefit.

How can cooperation evolve in a population of self-interested agents? This question has been addressed by numerous researchers [3,5,18,23]. My intention is not to develop a new answer to this question but rather to show that the previous answers are essentially the same: cooperation in all its forms can be viewed simply as a response by self-interested agents to their environment (which includes other agents) to increase their reward. Implicit in this view is the assumption that cooperation is indeed rewarded by the environment.

Consider, for example, an environment comprised of two selfish agents where each can decide to either reward or penalize the other. Once each has made its decision, the combination of decisions is evaluated in a *decision matrix* and each agent receives a *pay-off*. With the decision matrix in Figure 4a, the agents receive a higher pay-off for penalizing each other (competing) than for rewarding each other (cooperating). This situation might arise if the agents are competing for an non-sharable resource and are rewarded for the overall efficient use of the resource: if the agents fight for the resource, the resource will be used; if each is deferring to the other, then the resource will be wasted.[3] Cooperative agents will be at a disadvantage in this situation—in biological terms, a *selective* disadvantage, since more successful competitive agents will more likely survive. In a different scenario (Figure 4b),

[2]Intelligent agents can therefore intentionally communicate: they can communicate directly by exchanging messages, or they can communicate indirectly by performing actions on the environment that have the side-effect of transferring information when interpreted by other intelligent agents. In this paper, communication will be assumed to involve the exchange of information directly through messages.

[3]The agents are thus rewarded for finding a Pareto optimal allocation of the *indivisible* resource [53]. Such a reward scheme requires a single, global evaluator mechanism (Figure 1) to reward all agents.

AGENT-2

AGENT-1		Reward	Penalize
	Reward	0 / 0	1 / 1
	Penalize	1 / 1	2 / 2

(a)

AGENT-2

AGENT-1		Reward	Penalize
	Reward	2 / 2	1 / 1
	Penalize	1 / 1	0 / 0

(b)

The pay-offs for each agent given a combination of decisions is calculated by first locating the entry for the combination and then assigning the value in the lower left to agent-1 and the value in the upper right to agent-2. Thus, if both agents decide to reward, then matrix a would give each a pay-off of 0 and matrix b would give each a pay-off of 2.

Figure 4: Decision Matrices for Two-agent Environments.

the cooperating agents receive higher pay-offs and will have a selective advantage. This might occur if the agents want only part of a sharable resource: if each provides the remainder of the resource to the other agent, then more resource utilization will result than if one agent retains the entire resource but only uses part of it.[4] Cooperation will thus evolve in the second environment and not in the first precisely because the agents are self-interested *and* because cooperation pays.[5]

The prisoner's dilemma [3] represents an interesting two-agent environment (Figure 5). In general, self-interested agents should prefer to cooperate, since their average pay-off will be 3, as compared with an average pay-off of 2.5 for exploitation and 1 for competition. However, an agent that can consistently exploit other agents will receive a greater pay-off than if it cooperates. But other agents may not allow themselves to be consistently exploited. Hence the dilemma is not *why* the agents should cooperate (the pay-offs are inherent in the decision matrix) but rather *what* decisions should individual agents make to achieve cooperation and to avoid being exploited.

Axelrod identifies four attributes of agents that are successful in this environment [3]. First, the agents should be *nice* (they should always prefer cooperation). Second, the agents should be *provokable* (they should penalize agents that exploit them). Third, the agents should be *forgiving* (once they have penalized exploiting agents they should once again attempt to cooperate). Finally, the agents should be *clear* (their decisions should be predictable). With a diverse set of interacting agents, Axelrod found that an agent that gives TIT FOR TAT is most successful: it initially attempts to cooperate (nice), it

[4]The agents are rewarded for finding a Pareto optimal allocation of the *divisible* resource.

[5]This assumes that agents *rationally* pursue their perceived self-interests (try to maximize their payoffs). Individual preferences for outcomes in multi-agent domains can generally be represented as payoffs, although there are more detailed views of rationality in multi-agent environments [50,53].

		AGENT-2: Reward	AGENT-2: Penalize
AGENT-1	Reward	3 / 3	5 / 0
	Penalize	0 / 5	1 / 1

Figure 5: The Prisoner's Dilemma Decision Matrix.

responds to each penalty decision by another agent with exactly one of its own (provokable but forgiving), and other agents quickly learn that they cannot exploit it but they can cooperate with it (clear).

Axelrod's work illustrates how a population of cooperating agents can evolve and, once evolved, can stabilize. Biological researchers have shown that cooperating agents reinforce each other's survival and can display a hyperexponential growth curve—meaning that once cooperation has been established it cannot be eliminated [23]. Axelrod has found the same results in his computer simulations. But is it cooperation that evolves, or is it just the population of cooperating agents that evolves? That is, it appears that the trick to cooperation is in generating an environment where cooperation pays and in developing self-interested agents that can achieve this pay-off. Once these criteria are met, it does not seem so mysterious that the cooperating agents will have the selective advantage. Cooperation therefore occurs when, through chance or design, an environment exists where cooperation is advantageous and there exist agents that can capitalize on this environment.

2.4 Examples of Cooperation

Cooperation has been studied in many fields and by many researchers. This section briefly outlines how this simple view of cooperation as the fortuitous combination of appropriate agents in a favorable environment accounts for cooperation in a number of systems. In particular, cooperation has been cited in genetics, in connectionist research, in artificial intelligence, and in distributed computer systems. The discussion below presents a high-level view of cooperation in these systems; more rigorous studies are available in the references associated with each system.

Genes can be viewed as self-interested agents [18]; two genes cooperate when each increases the likelihood that both will be replicated (by generating a successful gene carrying plant or animal). For example, consider two genes that an animal might have: a teeth-type gene and an eye-placement gene.[6] Each of these may have two values: teeth-type may be "grinder" or "fang", and eye-placement may be "forward" or "separated". A carnivore needs forward eyes for hunting and fangs for killing, while a herbivore uses more separated eyes to better watch out for carnivores and has grinders for chewing plants. Since the other

[6]This simplified view of genes is consistent with Dawkins [18].

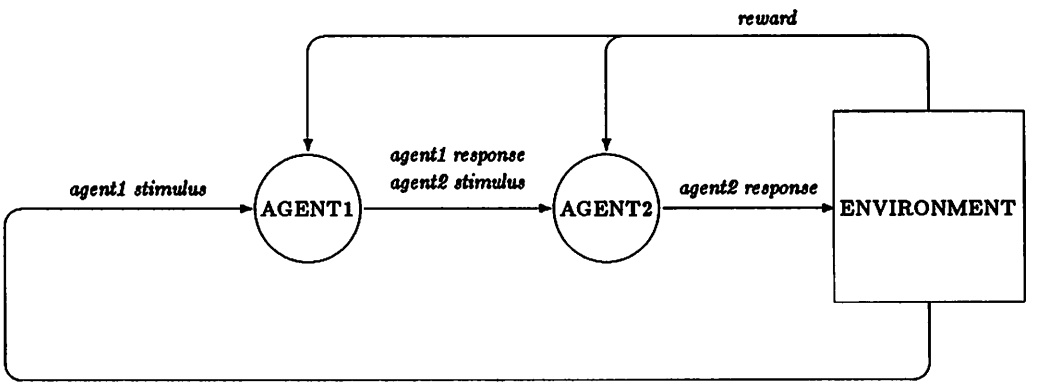


Figure 6: A Simple Network of Two Adaptive Agents.

two combinations will produce less viable animals (at least in our world), the successful combinations will replicate more often. Thus, unintentional cooperation between genes occurs both because the environment favors certain combinations of genes and because the random mixing of genes in a population can produce those combinations.

Neurons and neuron-like elements can be viewed as self-interested agents [5]. These agents have preferences for certain stimuli and are rewarded for receiving the desired stimuli. The agents adapt by modifying their responses based on the correlation between responses and rewards. For example, in a simple network (Figure 6, from Barto [5]) one agent receives stimulus from the environment and its response acts as stimulus to the other agent, which in turn responds to the environment. In essence, each agent is part of the other's environment. Cooperation occurs because the agents have compatible preferences since they receive the same rewards[7] and because they are able to generate responses that increase their rewards.

Distributed artificial intelligence systems promote cooperation by providing agents with appropriate goals or evaluation criteria that guide the agents into choosing cooperative interactions. For example, in systems that use negotiation to perform distributed problem solving [17], each agent has the goal of applying its local knowledge and computing resources to execute appropriate tasks (executing these tasks often generates tasks for other agents as well). The agents are self-interested (they want to satisfy their goals) but their self-interests lead them to cooperate to solve problems. As another example of distributed problem solving, the functionally-accurate/cooperative (FA/C) approach provides agents with shared high-level goals to achieve [39]. Each self-interested agent attempts to achieve these goals, and the agents tend to cooperate since they work on the same goals. Note, however, that if the agents have differing views on how to achieve these goals or different interpretations of these goals, then they may perform competitive actions. Examples of this phenomenon occur in human problem solving: although the president and the congress share the goal of lowering the deficit, their different perspectives might lead them toward competing solutions.

Rosenschein and Genesereth misleadingly call agents that share goals *benevolent* agents

[7] If their preferences were incompatible, the agents could not converge on a stable connection configuration.

[50]. Actually, these agents are completely self-interested since each performs actions only to satisfy its own local interpretations of these goals. Indeed, as observed above the sharing of goals is no guarantee that cooperative interactions will always occur. To escape what they perceive as a benevolent agent assumption, Rosenschein and Genesereth use decision matrix methods to study the prisoner's dilemma. They develop mechanisms that enable the self-interested agents to cooperate without explicitly sharing goals; their agents share the decision matrix instead. But since the purpose of both goals and decision matrices is to guide agents' decisions, agents that share decision matrices are no less benevolent than those that share goals.[8] In fact, intentional cooperation among intelligent agents requires that the agents have at least some information in common so that they can comprehend and anticipate each other's actions, but this common knowledge in no way forces them to be benevolent if it is not in their individual self-interests to cooperate.

As a final example, agents in a distributed computer system must often cooperate to effectively share network resources. For example, Kurose *et al* describe how selfish agents can cooperate to improve the use of a multiaccess communication channel [37]. If two or more agents attempt to transmit messages simultaneously, the messages collide and the transmissions are unsuccessful. Since agents are rewarded for successfully transmitting their messages, it is in their own best interests to avoid collisions. Kurose develops a microeconomic approach that allows each agent to cooperate by reducing its own rate of attempted transmission in return for other agents reducing theirs.

In conclusion, there are many examples of cooperative systems, but they all have two features in common. First, there must be potential benefits for cooperation in the agents' environment. In many cases, such an environment is purposely designed (a decision matrix is built, high-level goals are shared), while at other times cooperation is potentially advantageous by chance (such as in genetic evolution). The second feature is that agents must exist that can achieve these potential benefits. Once again, they can be intentionally designed (such as computer software) or created by chance (such as gene combinations and mutations). When these two events occur, self-interested agents will cooperate.

2.5 Cooperation Among Intelligent Agents

Intelligent agents must decide what actions they will take. While unintelligent agents may generate responses through knee-jerk reactions or random choices among alternatives, intelligent agents apply local knowledge to weigh alternatives and choose the best one. If this knowledge includes an explicit understanding of how different alternative actions might affect other agents, then an intelligent agent can purposely decide to perform cooperative actions. Alternatively, cooperation can be promoted implicitly by purposely providing a

[8] Attributes such as benevolence, malevolence, and altruism are conferred on an agent by other agents when the other agents cannot otherwise rationalize the agent's decisions. For example, an intelligent agent may perceive a parent's self-sacrifice as being altruistic, when in fact the parent may have had no choice (its actions are genetically ingrained and it cannot even recognize alternative actions). Altruism is attributed to the parent because the intelligent agent views the decisions differently. Similarly, an agent that appears to be benevolent (or malevolent) will only seem that way because other agents assign different rewards to its decisions than it does itself.

reward structure (for example, goals or a decision matrix) that causes each agent to rate cooperative actions highly.

The form that cooperation takes depends on the situation. Cooperative actions in one situation may be undesirable in other situations. For example, students may cooperate to improve each other's exam grades by discussing the material *before* an exam, but discussion *during* the exam is likely to achieve the opposite effect. Before the exam, the students cooperate by sharing resources (ideas and information), while during the exam the students should cooperate by avoiding harmful interactions (any interactions at all might lead to accusations of cheating).

Choosing appropriate actions therefore requires an understanding of the goals of cooperation. The generic goals of cooperation include:

1. To increase the task completion rate through parallelism.
2. To increase the set or scope of achievable tasks by sharing resources (physical devices, information, expertise, etc.).
3. To increase the likelihood of completing tasks (reliability) by undertaking duplicate tasks, possibly using different methods to perform those tasks.
4. To decrease the interference between tasks by avoiding harmful interactions.

Agents may be attempting to achieve one or more of these goals at any given time. In particular applications, specific variations and combinations of these goals of cooperation may determine how agents interact. For example, agents that are working together to solve a single problem may cooperate to achieve any number of goals. Associated with their related generic goals of cooperation (numbers in parentheses refer to generic goals above), these goals include:

- To increase the solution creation rate by forming subsolutions in parallel (1).
- To increase the variety of solutions by allowing agents to form local solutions without being overly influenced by other agents (1,4).
- To increase the confidence of a (sub)solution by having agents verify each other's results through rederivation using their potentially different expertise (2,3).
- To increase the probability that a solution will be found despite agent failures by assigning important tasks to multiple agents (3).
- To reduce the amount of unnecessary duplication of effort by letting agents recognize and avoid useless redundant activities (4).
- To improve the overall problem solving by permitting agents to exchange predictive information (2).
- To reduce the communication resource usage by being more selective about what messages are exchanged (4).

- To improve the use of computing resources by allowing agents to exchange tasks (2).
- To improve the use of individual agent expertise by allowing agents to exchange tasks (2).
- To minimize the time agents must wait for results from each other by coordinating activity (1,4).

Because all of these goals cannot be achieved simultaneously, agents that are working together to solve a problem must cooperate differently depending on the particular problem solving situation. For example, if a solution must be found quickly, the agents should not spend time verifying each other's results or developing a wide variety of solutions. Because of the diversity in the forms that cooperation can take in a distributed problem solving system, distributed problem solving is an appropriate context in which to study issues in cooperation.

2.6 Conclusion

A cooperating system can be viewed in two ways. When viewed as a single entity, the system decomposes problems and assigns subproblems to its various subcomponents. Alternatively, the system can be viewed as a collection of independent agents that can communicate. While the first view stresses intelligent *network* or *global control* (for appropriate problem decomposition and subproblem assignment), the second view emphasizes intelligent *local control* of each agent. The goal of this research is to study how cooperation can be achieved from this second perspective: how can independent, communicating agents make intelligent local control decisions that lead to effective cooperation.

In this section, I have developed the simple but important view of cooperation as agents taking advantage of an environment where suitable cooperation leads to increased rewards. The principal focus of the remainder of this paper will be on developing agents that are able to achieve the rewards of cooperation inherent in a distributed problem solving domain. In describing specific approaches and mechanisms for creating such agents, I rely on the conceptual foundation developed in this section. Specifically, I assume that there is no question about *why* the agents *should* cooperate in an environment that rewards cooperation through shared high-level goals, and concentrate instead on *how* the agents *can* cooperate effectively.

3. Distributed Problem Solving and Planning

An intelligent agent can apply knowledge to its current view of the environment and develop one or more possible actions to take. Since the agent is self-interested, it should choose the action that will reward it most, but this requires that the agent predict how its actions will affect the environment. If other agents will also be acting on the environment, the agent must attempt to choose an action that will suitably interact with other agents' actions to achieve a reasonable reward. If each agent's decision makes sense in the context

of the decisions of the other agents (allows each agent to achieve a reasonable reward), then the agents are cooperating coherently.

3.1 Distributed Problem Solving

Achieving coherent cooperation in a distributed problem solving network is a difficult task [17,39]. In a *distributed problem solving network*, each agent is a semi-autonomous problem solving *node* that can communicate with other nodes. Nodes work together to solve a single problem by applying their individual expertise to solve interacting subproblems and by integrating their subproblem solutions into an overall solution. These networks are typically used in applications such as distributed sensor networks [38,54], distributed air traffic control [7], and distributed robot systems [25], where there is a natural spatial distribution of information but where each node has insufficient local information to completely and accurately solve its subproblems.

Three important approaches have been taken to improve coordination among cooperating nodes: multi-agent planning, negotiation, and the functionally-accurate, cooperative approach.[9] In the multi-agent planning approach, the nodes typically choose a node from among themselves (perhaps through negotiation) to solve their planning problem and send this node all pertinent information. The planning node forms a multi-agent plan that specifies the actions each node should take and the planning node distributes the plan among the nodes. Since the multi-agent plan is based on a global view of the problem, all important interactions between agents can be predicted. If it can be effectively implemented, the approach seems suitable in domains such as air traffic control where it is imperative that node interactions (such as vehicle collisions) are assured of being detected and avoided [7]. Unfortunately, achieving a global view of the problem might be extremely costly both in communication resources and in time, and the performance of the entire network depends on the planning node and would be compromised if that node fails. The approach may thus be infeasible in many realistic situations.

In most domains, fortunately, the consequences of unexpected node interactions are not so dire; usually the interactions merely degrade performance. In the negotiation approach [17], a node will decompose a problem task into some set of subtasks and will assign these subtasks to other nodes (for parallel execution) based on a bidding protocol [54]. Since nodes may have different capabilities, the bidding protocol allows a subtask to be assigned to the most appropriate available node (nodes that are already working on subtasks are not available to bid until they have finished their tasks). A node that is awarded one subtask may thus be unavailable to perform a subsequently formed subtask despite being the best node for that subtask. If the node had been able to predict that a more appropriate subtask might soon be formed, the node would not have bid on the earlier subtask so that it would be available later. In addition, a node could use predictions about what nodes will be available when decomposing problems to form more appropriate subtasks. The inability of nodes to make predictions can therefore cause incoherence in the problem solving network:

[9] These approaches are by no means mutually exclusive [55], and an eventual goal of this research is to essentially combine aspects of each into a single framework.

the nodes could make a more coherent team and improve their overall performance if they could form more opportune subtasks and could assign subtasks to nodes better.

In the *functionally-accurate, cooperative* (FA/C) approach to distributed problem solving [39], nodes cooperate by generating and exchanging tentative, partial solutions based on their limited local views of the network problem. By iteratively exchanging their potentially incomplete, inaccurate, and inconsistent partial solutions, the nodes eventually converge on an overall network solution. To cooperate coherently, the nodes would need to predict what partial solutions would be exchanged in the future and when, so that they could modify their problem solving activities to form compatible partial solutions. To make these predictions, each node needs to understand its own plans and the plans of the other nodes. Without this understanding, nodes may require much more time to converge on a solution since they may work at cross-purposes.

Prediction is therefore crucial for coherent cooperation. While multi-agent planning requires accurate predictions before it can form acceptable plans, the negotiation and FA/C approaches can perform despite a lack of adequate predictions, but incoherence can degrade their performance. Better predictions in both of these approaches have been achieved through *organization*: by providing nodes with organizational information (the general capabilities and responsibilities of other nodes, the communication patterns between nodes), the nodes have a general understanding of each other and can therefore make better predictions. In the negotiation approach, this allows nodes to use focused addressing techniques to better form and assign subtasks to nodes, while in the FA/C approach, it allows nodes to better decide which of their potential problem solving tasks is likely to improve network problem solving. However, an organization can only make limited improvements to coherence since it helps a node predict *what* other nodes are generally likely to do, but provides little help in predicting *when* important interactions are likely to take place.

Coherent cooperation in a distributed problem solving network therefore requires that nodes predict when important interactions and events will occur and plan their local actions accordingly. By planning their activities, and by exchanging appropriate *meta-level* information about their plans, the performance of the entire network will improve.

3.2 Planners

By changing its environment, an agent can influence its future choices of possible actions, which in turn determine its future rewards. To increase its potential future rewards, therefore, an intelligent self-interested agent should *plan* to perform sequences of rewarding actions. For example, in a blocks-world situation, an agent may be rewarded for reducing the differences between the current block configuration and its goal configuration. However, reducing some differences may lead to situations where the other differences are more difficult to reduce. This is illustrated in Figure 7. The eventual *goal state* is to have block A on block B and block B on block C. In an *initial state* with block A on block C, the agent could move block A either onto block B (*state 1*) or onto the table (*state 2*). Since putting it on B achieves part of the goal state (A on B), it appears to be the most rewarding short-term action. However, putting B on C requires that both blocks be clear: to achieve

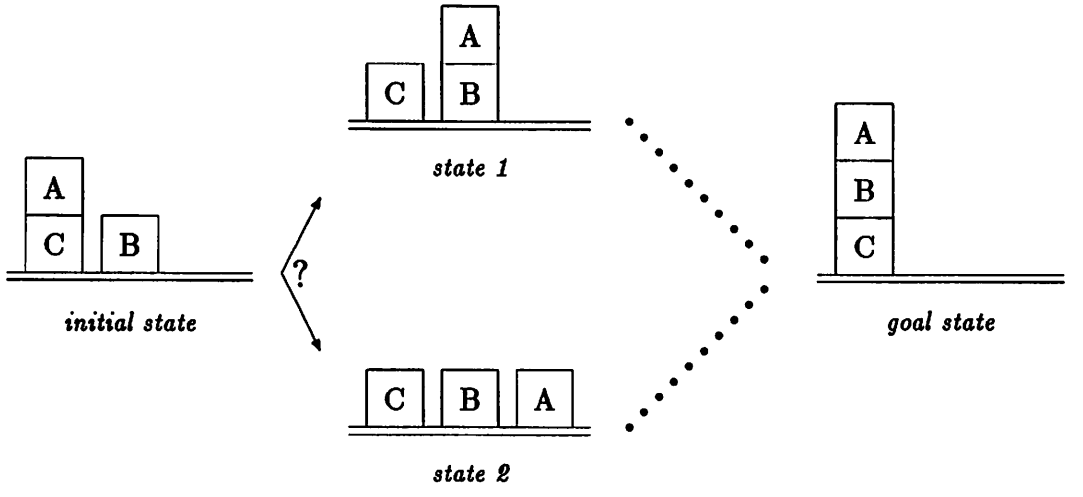


Figure 7: A Blocks-World Planning Decision.

the long-term rewards, block A should be placed on the table. For the agent to take the correct action it must be able to predict its future environments; it must be capable of simulating the effects of actions on an internal model of its environment. By doing so, the agent forms a plan (a sequence of appropriate actions) to achieve its goal.

3.2.1 Planning and Execution

A plan's effectiveness depends on how it performs in the actual environment. If the agent has a complete and certain model of its environment and actions, then the plan should achieve the same rewards as the agent had predicted. The simplest type of planner assumes that the only changes to the environment are caused by a single agent and that the actions of this agent have completely predictable effects on the environment. In essence, such a planner assumes that the actual environment always behaves exactly as the internal model of the environment does. Many early planners such as STRIPS were based on this assumption [27]. Even with this assumption, planning can be a complex problem because of the potential interactions between goals (or subgoals) to be achieved. The earlier blocks-world example illustrated this: by deciding to achieve one of its goals (A on B) before the other, an unsuitable plan can be developed. Later planning systems, such as INTERPLAN [57] and Waldinger's system [61], reordered goals to avoid harmful plan interactions; other planners, like NOAH [51] and NONLIN [58], only imposed an order on goals when interactions between plans to achieve the goals independently indicated that a particular ordering was necessary. In all of these systems, plans were formed assuming that the environment and actions could be modeled with complete certainty, so that choosing a plan step requires the planner only to explore the ramifications of its possible actions.

When the effects of an action cannot be predicted with certainty, or when events beyond the agent's control can change the environment, then the planner cannot assume that the plan it forms will necessarily be successful (the distinction between actions and events was

made Section 2.2). Typically, planners in such situations will develop a plan based on their internal models but will monitor the actual execution of the plan to detect deviations between what was expected and what actually occurred. When the plan deviates, the planner will alter the remainder of the plan to account for the unanticipated environment, and the agent continues its activity (again with the plan being monitored). For example, NOAH develops a plan representation that facilitates execution monitoring and replanning [51], and Wesson describes a system that monitors and revises plans [62].

Sacerdoti has recognized that more advanced planners should integrate plan generation, execution, and repair [52]. In unpredictable and uncertain environments, it may be pointless to plan actions for the distant future [9,16,26,43]. The planner should adopt a "wait and see" approach in these situations by only planning as far in the future as it can reasonably predict. After executing its partial plan (alternatively, its plan to achieve some set of subgoals), the planner can reconnoiter before developing its next partial plan. By interleaving planning and execution, the planner wastes less time forming plans that are unlikely to come to fruition; by maintaining a view of its overall goals, the planner can choose an effective sequence of partial plans to achieve these goals. Monitoring and repair are still needed since the partial plans may themselves deviate from expectations.

3.2.2 Multi-agent Planning

Planning is complicated when more than one agent can affect the environment. A *multi-agent* planner develops a plan that specifies the actions that each agent should take. For example, if several agents each want temporary access to some non-sharable resource, the multi-agent planner generates a plan where each gets its turn without interfering with the others (for example, see Wilkins [63], Georgeff [30], and Corkill [10], and Fox on multi-agent scheduling [29]). A multi-agent planner therefore coordinates the actions of the agents to achieve the maximum reward for the agents as a group. In an environment that rewards cooperation between self-interested agents, the multi-agent planner maximizes these rewards.

Constructing multi-agent plans in a distributed environment is a complex problem. A typical approach is to designate one of the agents as the planner: this agent collects all of the relevant information from the other agents, forms a multi-agent plan, and then distributes this plan among the agents. This *centralized* approach has been used, for example, in air traffic control [7], but though straightforward, may require large amounts of communication and may reduce agent responsiveness since agents must wait while the multi-agent plan is formed and distributed. Since planning can be information and computation intensive, centralized multi-agent planning in environments with slow and unreliable communication may be impractical and will certainly be wasteful of computation resources since all but the planning agent sit idle.

In distributed multi-agent planning, agents concurrently form their own partial plans, which together comprise the complete multi-agent plan. Since no single agent has a global view of the agent actions, detecting and dealing with interactions among agents is much more difficult. The general approach is to provide each agent with a model of other agents

or other agents' plans[10], (for example, Corkill's MODEL nodes [10], Georgeff's process models [31], Konolige's belief subsystems [36], Rosenschein and Genesereth's exchange of tentative multi-agent plans [49]). Because of communication bandwidth limitations and communication delays, agents might have incomplete, inaccurate, and inconsistent models of other agents, and an agent must be capable of anticipating plan interactions despite these difficulties. Plan interactions are usually reconciled by having agents synchronize *critical regions* of the plans [10,30]).

In an unpredictable environment, the performance of a multi-agent plan should be monitored and replanning should be performed when necessary. Plan monitoring and revision are complicated in distributed systems as well. A centralized monitor could observe the entire network and could halt and revise plans as needed, but this approach would require vast amounts of communication. Alternatively, each agent could monitor its own part of the plan and either replan locally or call for an entirely new multi-agent plan, but some execution errors cannot be detected locally (such as a lost synchronization message or a fatal combination of locally acceptable errors). A hybrid approach would allow agents to exchange acceptably small amounts of information among themselves to update their models of each other so that non-local execution errors could be detected. By replanning locally and informing other agents of relevant changes to their plans, agents could respond to unexpected situations in a decentralized manner.

3.2.3 Planning Using Time

Most planners do not associate actions with their durations. In these planners, the order of plan steps defines the temporal relationships between actions, and predictions about "when" an action will occur can only be expressed in terms of the actions that must precede it. In a multi-agent environment, agents are therefore unable to predict what actions will be occurring simultaneously in the network. As described above, coordination between agents is enforced through specific synchronization actions (communications) that allow agents to proceed with their plans only when other agents have completed actions in a certain critical region.

If agents were able to represent the temporal aspects of plans, then they could predict when actions would occur. A multi-agent plan that incorporated such knowledge would not need to use synchronization primitives: the actions of each agent could be timed to avoid harmful interactions and to promote beneficial interactions.[11] Cheeseman describes a multi-agent planner that uses temporal reasoning to avoid resource conflicts and deadlocks [8]. Other research in planning has incorporated temporal reasoning into planners [1,42,44,60]. For example, these planners can develop plans that allow agents to prevent the predicted actions of other agents from occurring [1], or to base planning steps around

[10]The same approach has been used in natural language (story) understanding systems for interpreting the actions of various agents (characters). For example, Bruce and Newman used belief spaces to represent agents models of each other [6]. Economic planning also involves issues in information flow for decentralized planning [34].

[11]This assumes that small variations between agents' local clocks are negligible relative to task durations.

expected events [60].

If the duration of planning actions cannot be predicted accurately, the planner has several alternatives. To coordinate critical interactions, communication between agents to synchronize their actions is a suitable method to ensure coordination. When perfect coordination is not essential, agents might proceed with their individual plans despite the possibility of interacting inappropriately. Finally, to improve coordination without resorting to explicit synchronization, agents could occasionally inform each other of their progress and dynamically modify their plans. For example, agent-1 and agent-2 plan to simultaneously execute action-1 and action-2, respectively. If agent-1 is informed that agent-2 has been delayed during earlier actions in its plan, then agent-1 could modify its plan, possibly by inserting useful new actions ahead of action-1. In this way, agent-1 is not sitting idle while waiting for a synchronization message, but instead is responding flexibly to its unpredictable environment.

3.2.4 An Integrated Planner

To perform appropriately in a complex distributed problem solving environment, an agent's planner must integrate the attributes of the planners described above. First, because the problem characteristics can change as problem solving progresses, agents should monitor and revise their plans, and should interleave planning and execution (they should not develop complete and detailed plans when those plans are unlikely to be useful in the long run). Second, agents should develop and maintain models of each other so that they can create multi-agent plans in a distributed manner and can make local modifications to these plans in response to unexpected situations. Third, the agent must be capable of temporal reasoning so that it can predict and plan around important actions and events. Its models of other agents allow it to predict their future actions; if it also has a model of the environment, it may be able to predict future events (a meteorologist can predict the weather despite being unable to affect it).

Since an agent's models of other agents and of the environment may be incorrect, it may make incorrect predictions or fail to make some important predictions. Agents may exchange information (within the communication resource constraints) to improve each other's predictions, but with improved predictions an agent may replan and thus invalidate other agents' predictions about its actions. Unlike typical multi-agent planners that ensure effective cooperation while a plan remains valid, the integrated planner allows agents to make incorrect decisions because of their potentially incomplete, inaccurate, and inconsistent knowledge. However, this planner is more responsive and more generally applicable than a typical multi-agent planner: agents can respond to unexpected changes in their environments by replanning locally and by exploring alternatives (rather than ceasing their actions until a new multi-agent plan is formed); and this approach can be used in domains where communication is severely restricted (although increasing the exchange of pertinent information will generally improve cooperation since agents' models of each other can be more up-to-date).

In conclusion, planning in complex domains is a complicated problem, requiring planners that can reason about time, can interleave planning and execution, can monitor and

adapt their plans, and that can predict when important actions and events will occur that might affect plans. In the next section, I describe an implementation that incorporates a preliminary version of such a planner into a testbed for simulating a distributed problem solving network. This planner can predict the actions of a problem solving node in the near future, and mechanisms are described that allow nodes to exchange *meta-level* information about these predictions. The experimental results based on these mechanisms are promising, and as a result, the subsequent section outlines proposed research toward more advanced mechanisms that improve the nodes' abilities to predict important actions and events, to form plans that use these predictions, to exchange information to improve cooperation, and to respond to unexpected actions and events intelligently.

4. Preliminary Research

Coherent cooperation is difficult to achieve in a distributed problem solving network because each problem solving node has only limited, possibly incorrect views of the activities of other nodes. By combining these views with a view of its local activities, a node forms a model of overall network activity. This model guides the node's local decisions about what tasks to perform and what messages to send and receive. Therefore, the degree to which nodes cooperate coherently depends on the appropriateness of this model of network activity.

The first step in developing a useful model of network activity is to provide each node with the ability to form a useful view of its local activity. This may be much more difficult than it sounds. Developing such a view requires the node to find relationships between past, present, and possible future actions so that it forms a high-level perception of its activities. An opportunistic node that explores various solution possibilities, that attempts to solve diverse goals in parallel, and that responds to new information quickly, may have difficulty in finding these high-level relationships since it jumps from task to task based on their individual ratings rather than pursuing multi-step sequences of related tasks.

Once these relationships have been found, the second step in developing a useful model of network activity is to give nodes the ability to determine what aspects of their views of local activities might be important for other nodes to know. To improve coherence, each node should determine how its own actions will affect network activity, but to make this determination a node may need to know only certain relevant information about specific nodes. Therefore, improving models of network activity should not be accomplished by having nodes broadcast detailed views of their activities—such an approach not only requires excessive communication resources but will also force each node to extract the important information before it can make intelligent modifications to its model. Nodes should have sufficient knowledge to enable them to make informed decisions about what aspects of their high-level views should be exchanged.

Preliminary mechanisms to achieve some of these capabilities have been implemented in an experimental testbed that simulates a distributed problem solving network. The implementation and results have been thoroughly discussed elsewhere [20]. The remainder of this section presents a brief synopsis of the testbed, the mechanisms, and the experimental

results based on the mechanisms.

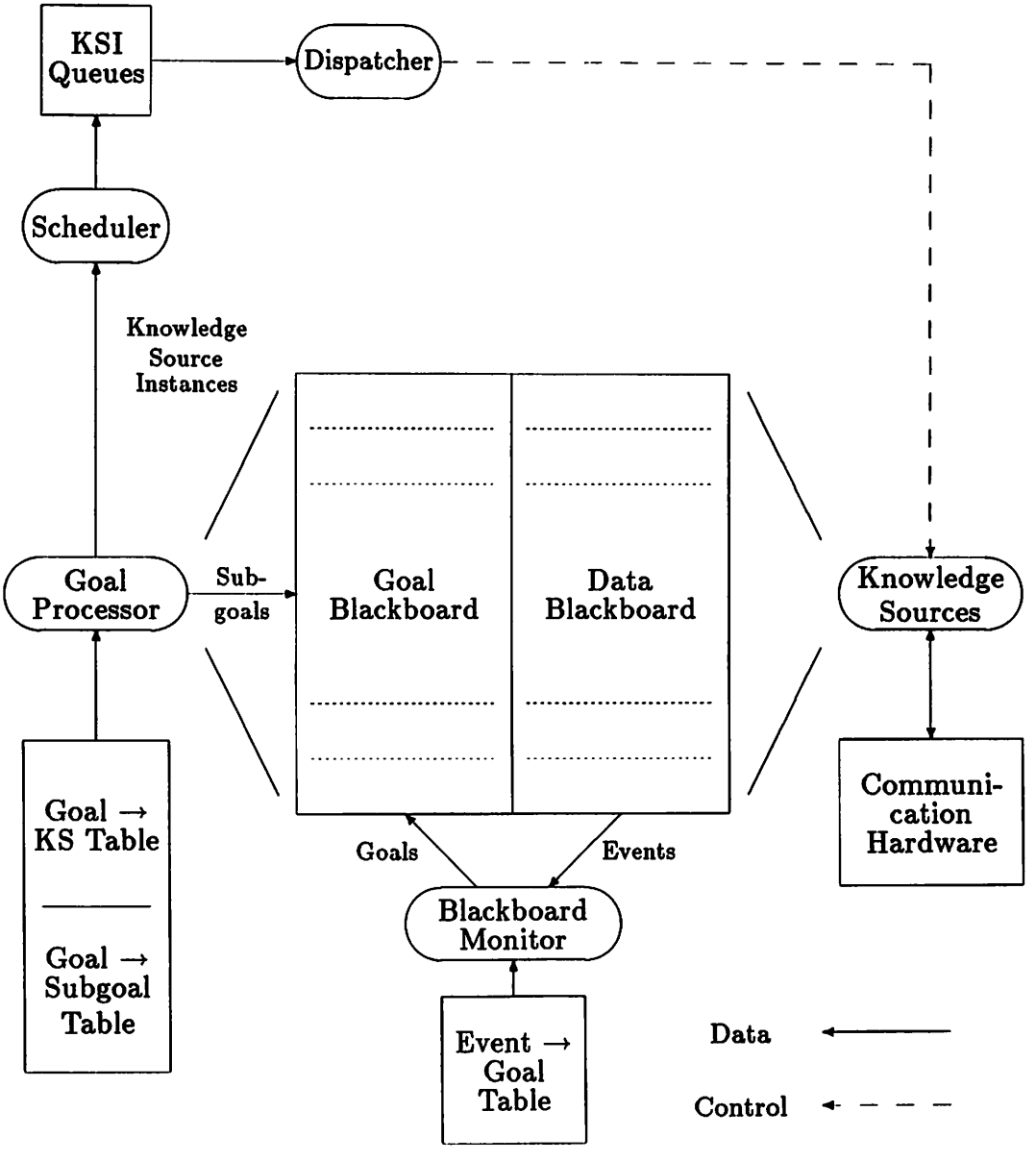
4.1 The Experimental Testbed

By simulating a network of problem solving nodes, the Distributed Vehicle Monitoring Testbed (DVMT) provides a framework where general approaches for distributed problem solving can be developed and evaluated [40,41]. Each simulated node applies its own simplified signal processing knowledge to acoustically sensed data in an attempt to identify, locate, and track patterns of vehicles moving through a two-dimensional space. By varying parameters in the DVMT that specify the accuracy and range of the acoustic sensors, the acoustic signals that are to be grouped together to form patterns of vehicles, the power and distribution of knowledge among the nodes in the network, and the node and communication topology, a wide variety of cooperative distributed problem solving situations can be modeled.

Each problem solving node has a Hearsay-II, blackboard-based architecture [24], with knowledge sources and levels of abstraction appropriate for vehicle monitoring. A *knowledge source* (KS) performs the basic problem solving tasks of extending and refining *hypotheses* (partial solutions). The basic Hearsay-II architecture has been extended to include more sophisticated local control and the capability of communicating hypotheses and goals among nodes [12,13]. In particular, a goal blackboard, a goal processing module, and communication knowledge sources have been added (Figure 8).

Goals are created on the goal blackboard to indicate the node's intention to abstract and extend hypotheses on the data blackboard. Through goal processing, a node forms *knowledge source instantiations* (KSIs) that represent potential KS applications on specific hypotheses to satisfy certain goals. The scheduler ranks a KSI based both on the estimated beliefs of the hypotheses it may produce and on the ratings of the goals it is expected to satisfy. Appropriate goal processing to modify goal ratings can therefore alter KSI rankings to improve local control decisions. The network organizational structure, for example, is specified as a set of node *interest areas* that affect goal processing decisions: a node will increase the ratings of goals that fall within its areas of network responsibility while decreasing the ratings of other goals. Since goal ratings affect KSI rankings, the interest areas can influence node activity; but because there are other factors in ranking KSIs (such as the expected beliefs of the output hypotheses), a node still preserves a certain level of flexibility in its local control decisions. The organizational structure thus provides guidance without dictating local decisions, and can be used to control the amount of overlap and problem solving redundancy among nodes, the problem solving roles of the nodes (such as "integrator", "specialist", and "middle manager"), the authority relations between nodes (whether nodes are biased to prefer work based on received data or on locally generated data), and the potential problem solving paths in the network [15].

Put simply, a node repeatedly performs a problem solving cycle: it executes a KS to post hypotheses on the blackboard, the new hypotheses trigger the creation of suitable goals for refining and extending them, the goal processing forms and rates KSIs to satisfy these goals, and the scheduler chooses the KSI to invoke next and triggers the appropriate



The principal components of the problem solving architecture of a DVMT node are illustrated.

Figure 8: DVMT Node Architecture

KS to execute. Concurrently, a transmission processor and a reception processor each maintains a queue of communication KSIs (ranked by the rating of the information to be communicated and by the interest areas), and each executes KSs to transmit or receive messages [19]. The cyclic problem solving activity begins when low-level signal hypotheses are posted based on sensor data, and terminates when one or more high-level hypotheses matching a predefined solution is formed.[12]

4.2 Preliminary Mechanisms

An organizational structure guides nodes into improved cooperation by limiting the possible interactions between nodes. However, to cooperate even more coherently, nodes must be able to plan their activities and anticipate when interactions will occur. The principal mechanisms for achieving more coherent cooperation are a planner to improve a node's self-awareness and meta-level communication to allow nodes to exchange information that will improve their models of network activity (improve their network awareness).

A *plan* represents a desire to achieve a high-level goal by performing a sequence of activities (KSIs). To identify plans, the node needs to recognize these high-level goals. Inferring high-level goals based on pending KSIs is an inappropriate approach; it is like attempting to guess a chess opponents strategy by looking at isolated moves. Furthermore, the hypothesis and goal blackboards provide information at too detailed a level to efficiently infer these high-level goals. What is required is a structure similar to the blackboards where related hypotheses and goals are grouped together. A preliminary version of this structure, called the *abstracted blackboard*, is a structure reminiscent of the focus-of-control database first used in the Hearsay-II speech understanding system [24,32].

Hypotheses with similar level, time, and region characteristics are grouped together on the abstracted blackboard. This grouping acts as a smoothing operator, obscuring details about individual hypothesis interactions so that broader, long-term interactions between areas of the solution space can be discerned. By transforming the data blackboard into the abstracted blackboard, we explicitly generate a state representation that is uniquely appropriate for planning at a higher level of abstraction.

In the preliminary implementation [20], the abstracted blackboard takes the form of a two-dimensional array, with level and time indices. When a hypothesis is created, its level and time attributes are used to identify the level-time entries in the abstracted blackboard to be modified. Each level-time entry contains some number of regions, and the new hypothesis is either incorporated into an existing region (if it falls within the region) or causes a new region to be formed. The level-time-regions of the abstracted blackboard are each summarized into a set of values that are derived from the associated hypotheses. These values include the maximum belief of the hypotheses in the level-time-region, the

[12]The solution is thus specified before problem solving begins, but a node cannot use this information to guide its processing. Without the availability of such an "oracle", termination of problem solving is much more difficult, requiring nodes to determine whether further problem solving is likely to improve upon a possible solution which it has already generated or will cause it to generate another potentially better solution. In effect, the termination decision depends on the criteria for deciding whether network goals have been satisfied [14].

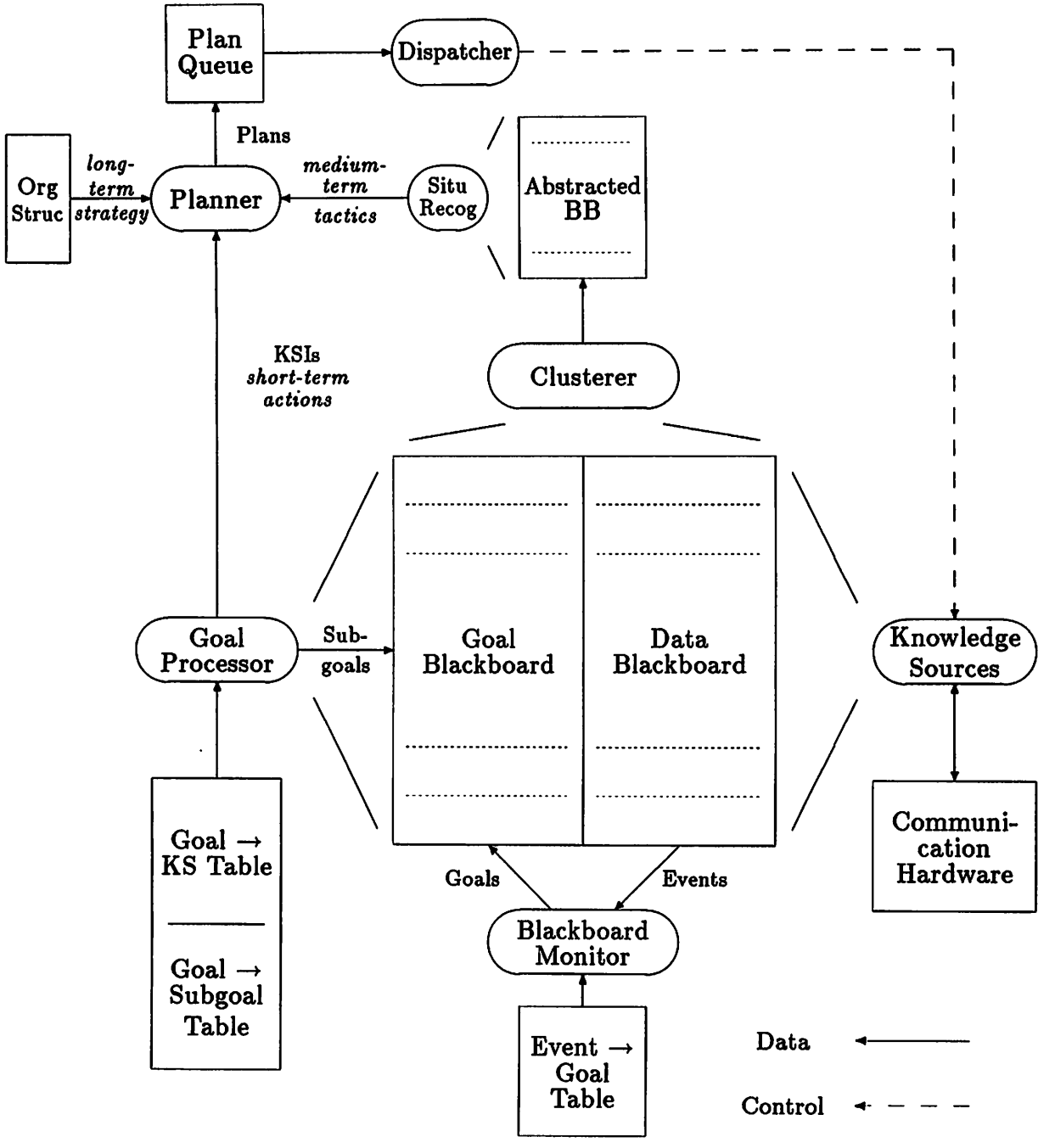
number of highly believed hypotheses, the number of KSIs stimulated by these hypotheses that have yet to be invoked, the total number of hypotheses in the level-time-region and how many uninvoked KSIs are associated with them, and an indication as to the other level-time-regions that share at least one of the hypotheses.

A *situation recognizer* scans the abstracted blackboard to develop a higher level view of the problem solving occurring at the node. For example, low maximum belief indicates the problem solving approach in that area should be re-evaluated, a large number of equally rated hypotheses could imply that there is uncertainty that should be resolved, and a large number of pending KSIs indicates the need for making an informed and judicious choice as to which action to take next. Based on this view, it generates higher level goals and passes them to the planner. A goal might be to extend a highly believed portion of the blackboard into a new area or to improve credibility in another area. The *planner* in turn creates a plan to satisfy the higher level goal, and associates with the plan a set of KSIs that represent potential steps in the plan. This set is updated as the plan is carried out and appropriate new KSIs are formed.

At any given time, a node will work on its most highly rated plan; problem solving is opportunistic because plans are interruptable. The numeric plan rating is computed from several factors, including the rating of the high-level goal (the maximum belief of the data it encompasses), the interest-area specifications for that portion of the solution space, the rating of the best pending KSI associated with the plan, and whether the high-level goal is based on data formed locally (to reason more fully about potentially distracting information received from other nodes). The control structure of a node has thus been modified (Figure 9). As the figure indicates, the creation and ranking of plans requires the planner to integrate the influences of the long-term strategy of the organizational structure (the interest areas), the medium-term higher-level view of the current situation (the abstracted blackboard), and the short-term KSI input indicating actions that can be achieved immediately (the KSI queue). Hence, decisions in a plan-based node are more informed than those in a KSI-based node (a node without plans).

Although the planner can improve a node's local decisions, these decisions may still be incorrect because they are based only on a node's local view of network activity. To improve a node's network awareness, the second mechanism, meta-level communication, is introduced. Meta-level information about the past, present and future actions of the nodes is specifically intended to increase coherent cooperation in the network. Abstracted blackboard entries summarize a node's hypotheses, and thus provide a high-level view of a node's past actions. By exchanging portions of the abstracted blackboards, nodes can reason about the *past* activities of their neighbors. Furthermore, a node that knows the current plan of its neighbor can reason about its *present* actions. Reasoning about the *future* actions of a node, however, is a complex problem. Not only must estimations be made about the duration and effects of a node's plans, but also about what further information the node may receive (from another node or from its sensors) that could affect these estimations. This is an issue that will be studied in the proposed research.

The preliminary implementation assumes that a node can make completely accurate short-term predictions about future activity based solely on the plan queue. We simu-



The principal components of the modified problem solving architecture of a DVMT node are illustrated.

Figure 9: The Modified Problem Solving Architecture of a Node.

late this best-case scenario by letting nodes access the abstracted blackboards and plan queues of other nodes. When developing a plan, a node can use this additional meta-level information to decide where to focus the node's problem solving activity; for example, it can determine if the plan will redundantly derive information that another node has either generated (present in the abstracted blackboard) or is in the process of generating (the top plan). By avoiding redundant activity, solution generation rate can improve because less highly rated but potentially useful activities will be invoked earlier (rather than redundant invocation of highly rated activities). Meta-level information also allows nodes to better predict the effects a message will have on other nodes; this additional information can be used by the communication processors to alter communication KSI ratings, thereby improving communication decisions [22].

4.3 Preliminary Results

The planner and meta-level communication mechanisms were implemented and evaluated on a large number of environments [20]. The following is a brief summary of the experimental results and the conclusions that can be drawn.

The planner can significantly improve problem solving performance of individual nodes because the nodes can recognize and reason about entire sequences of KSIs. In degenerate one-node networks, the problem solving performance is often optimal (the node made only correct problem solving decisions). When significant amounts of noisy data are present, the planner may make incorrect decisions but will recover from these decisions quickly thanks to its high-level view of node activity and goals. Networks of plan-based nodes (each node has a planner) perform better because each node performs better individually. However, although networks of plan-based nodes had improved performance, the nodes occasionally made incorrect decisions because they did not anticipate interactions with other nodes.

By exchanging meta-level information, the nodes can make more informed local decisions based on both their high-level view of local activity and on received high-level views of activity at other nodes. As a result, nodes can avoid unhelpful interactions such as duplicating each other's problem solving activities, and network problem solving performance improves. In fact, the experiments stressed issues in avoiding redundant problem solving actions by nodes, and the planner and meta-level communication mechanisms allowed all such redundancy to be avoided. The planner, when deciding whether to focus the node's processing in a certain area could examine the meta-level information from other nodes to determine what results, if any, had been found for that area.[13] It would choose to work in that area only if it was the first to do so, or if its data (as summarized in the abstracted blackboard) were superior to that of any predecessors.

By modifying the planner's response to meta-level information, the mechanisms could similarly help a node to achieve other goals of cooperation: if it wished to corroborate results using its own particular expertise, it would choose to work in areas where other nodes had previously worked; if it wished to avoid distracting other nodes, it could re-

[13]Since nodes might only exchange high-level hypotheses, there may be a long delay between when work on an area begins and when hypotheses about that area are exchanged.

frain from sending results that it believes (based on the abstracted blackboard) could be combined with their local data; if it wished to guide other nodes, it might purposely send such results. Thus, given some (possibly evolving) knowledge about the goals of cooperation, the planning and meta-level communication mechanisms can guide nodes into making better "team" decisions. Similar mechanisms could be used in other domains where an agent must recognize its alternatives and choose the alternative that makes the most sense considering the actions of other agents and the goals of the group.

Planning and meta-level communication introduce additional computation and communication overhead. To discover when the additional overhead is worthwhile in the DVMT, a sequence of environments were simulated where duplication of effort was unnecessary (did not influence confidence) and where overlap between nodes was varied. Planning and meta-level communication could improve network performance by guiding nodes away from performing highly rated but redundant work and toward lower rated useful work sooner. Consequently, the network could find a solution in less time. In environments with little or no overlap, the opportunities for redundancy were so limited that the overhead of the new mechanisms essentially negated the modest reduction in redundancy. However, as the environments became more complex because of increasing overlap, the planner and meta-level communication became indispensable: despite their overhead, these mechanisms improved performance by drastically reducing the number of incorrect problem solving decisions that nodes made.

In conclusion, the planner and meta-level communication can increase network coherence. In fact, in the types of environments that were studied and with the assumptions that were made, these mechanisms often made network problem solving completely coherent. However, these mechanisms need more work before they can become applicable in a wider range of environments with different problem solving situations and network characteristics. For example, in very complex (overlapping) environments, the exchange of obsolete meta-level information degraded performance—the network could have done better without any meta-level communication at all. In the next section, I propose research that will improve these mechanisms so that they can be more widely applicable.

5. Proposed Research

The emphasis of the preliminary research was on building mechanisms that enabled nodes to avoid unnecessarily duplicating each other's work. The implicit assumption behind these mechanisms was that the independent derivation of the same hypothesis by more than one node did not increase the confidence in that hypothesis since nodes had identical problem solving expertise (except in hierarchical organizations). With this assumption, the goal of cooperation is to reduce the amount of redundant processing since it wastes computing resources that may be better spent deriving new hypotheses.

In other situations, as described in Section 2.5, the goal of cooperation may be different. To recognize how their self-interests can be furthered through cooperation, the nodes must be able to recognize and react to specific aspects of their environment. The principal thrust of the proposed research will be to develop mechanisms with the versatility to achieve the

various goals of cooperation. It will therefore be necessary to form a taxonomy of these goals. The mechanisms will be implemented in the framework of the DVMT, and the implementation will be extensively tested on environments that embody the different goals of cooperation. Finally, the potential uses of these mechanisms in other distributed problem solving systems and more general distributed computing applications will be studied.

5.1 Implementation and Evaluation

Although the preliminary mechanisms described earlier provide evidence that planning and meta-level communication are appropriate approaches for improving cooperation, the mechanisms were primitive and limited. The main focus of the proposed research will be the implementation of more sophisticated and versatile mechanisms: an abstracted blackboard that permits a large repertoire of problem solving situations to be recognized; a planner that can explicitly represent sequences of actions in response to the situations, that can predict when certain results will be generated and what changes to the environment might cause the plan to be abandoned, and that can anticipate actions of other nodes based on potentially obsolete, inaccurate, or incomplete meta-level information; meta-level communication mechanisms that promptly send only the most important meta-level information (within communication resource limitations); and network control mechanisms that allow tasks and responsibilities to be exchanged between nodes.

5.1.1 Planning and Prediction

A plan represents a prediction about future activity. Although uncertain about future events and other nodes' actions, a node that has decided to completely perform a plan (sequence of actions) can predict exactly the actions it will be taking.[14] A node that has determined to completely perform a plan has *committed* to that plan [1,28]. If each node sends information about the plans to which it has committed, then each node could accurately predict the relevant actions of other nodes and could therefore predict its important interactions with those nodes. Predictability improves the coherence of cooperation since nodes can anticipate when specific helpful and harmful interactions will occur.

By committing to a plan, however, a node reduces its responsiveness to changes in the environment caused by events beyond the node's control. An unexpected event can alter the environment so that the current plan appears less worthwhile compared to other possible plans. For example, a node may be executing a plan to attempt to form a certain result when new evidence arrives indicating that an alternative result may in fact be more likely. Now the node is in a quandary: should it continue or abandon its current plan. If it maintains its commitment to its plan, then it will cooperate smoothly with other nodes but may waste its time generating an unimportant result (although because of the uncertainty

[14] It cannot, however, necessarily predict the effects of those actions. For example, if the plan is to execute a sequence of KSs, the KSs may create different hypotheses than expected. Although the actions (KSs) taken are predictable, the results are not. In an automated factory, as another example, a robot that is instructed to tighten a screw can go through the tightening motions even if another robot has unexpectedly taken the screw.

involved in predicting plan outcomes, the result could be useful). On the other hand, if it abandons the current plan in favor of a new one, then it may potentially form a more useful result (although this is by no means guaranteed) but could disrupt the cooperative interactions between nodes.

The planner must consider tradeoffs between predictability and responsiveness. At times, continuing a potentially sub-optimal plan may result in better network performance due to improved cooperation, while at other times network problem solving will benefit from a node responding to new information despite a resulting degradation in cooperation. For example, $node_1$ may initially use an abstract view of its data to recognize that there are two possible alternative tracks in its sensed area, and forms a plan for each possibility ($plan_1$ and $plan_2$). It begins executing $plan_1$ (which it rates more highly), and informs its neighbors of this plan. After pursuing $plan_1$ for a while, it receives meta-level information from $node_2$, informing $node_1$ of its plan to form a complementary track to combine with the results of $plan_1$. Next, $node_1$ gets a partial result from $node_3$ that makes $plan_2$ look more promising than $plan_1$. At this point, $node_1$ could abandon $plan_1$: if $plan_2$ lives up to its promise, this decision might be appropriate but will definitely cause $node_2$ to waste its time; if $plan_2$ fails to meet its promise, $node_1$ will waste time before returning to $plan_1$. On the other hand, $node_1$ could continue with $plan_1$: $node_1$ and $node_2$ will cooperate smoothly to form a combined track, but this track may or may not be the best track possible. The planning mechanism, thus, must not only be able to form appropriate plans for the different problem solving situations the node might encounter, but must also be able to make intelligent decisions about when to abandon a plan.

The first step in constructing such a planner is to develop mechanisms that allow the node to recognize a variety of problem solving situations. The preliminary implementation of the abstracted blackboard was a first step toward this end, but had serious limitations. Since hypotheses were grouped based on a hardwired criterion, the grouping algorithm biased the high-level view that was achieved. The representation was an inflexible array structure that would not allow more interesting clustering of hypotheses. Finally, the situation recognition algorithm was hardwired to look for important situations in a predefined order with the goal of reducing unnecessary redundancy of effort between nodes.

I propose to develop more versatile mechanisms for building and representing a node's abstracted blackboard. Already partially developed, these mechanisms allow a node to cluster hypotheses differently depending on the situations it *expects* to find. Thus, bias remains an important aspect of the abstraction mechanisms—without bias a node could not choose between different ways to cluster hypotheses—but the bias is represented explicitly.[15] If the expectations of a node change as a run progresses (perhaps based on the clusters it forms), the bias can be altered to reflect the new expectations. Bias could also be affected by information received from other nodes: since bias affects a node's

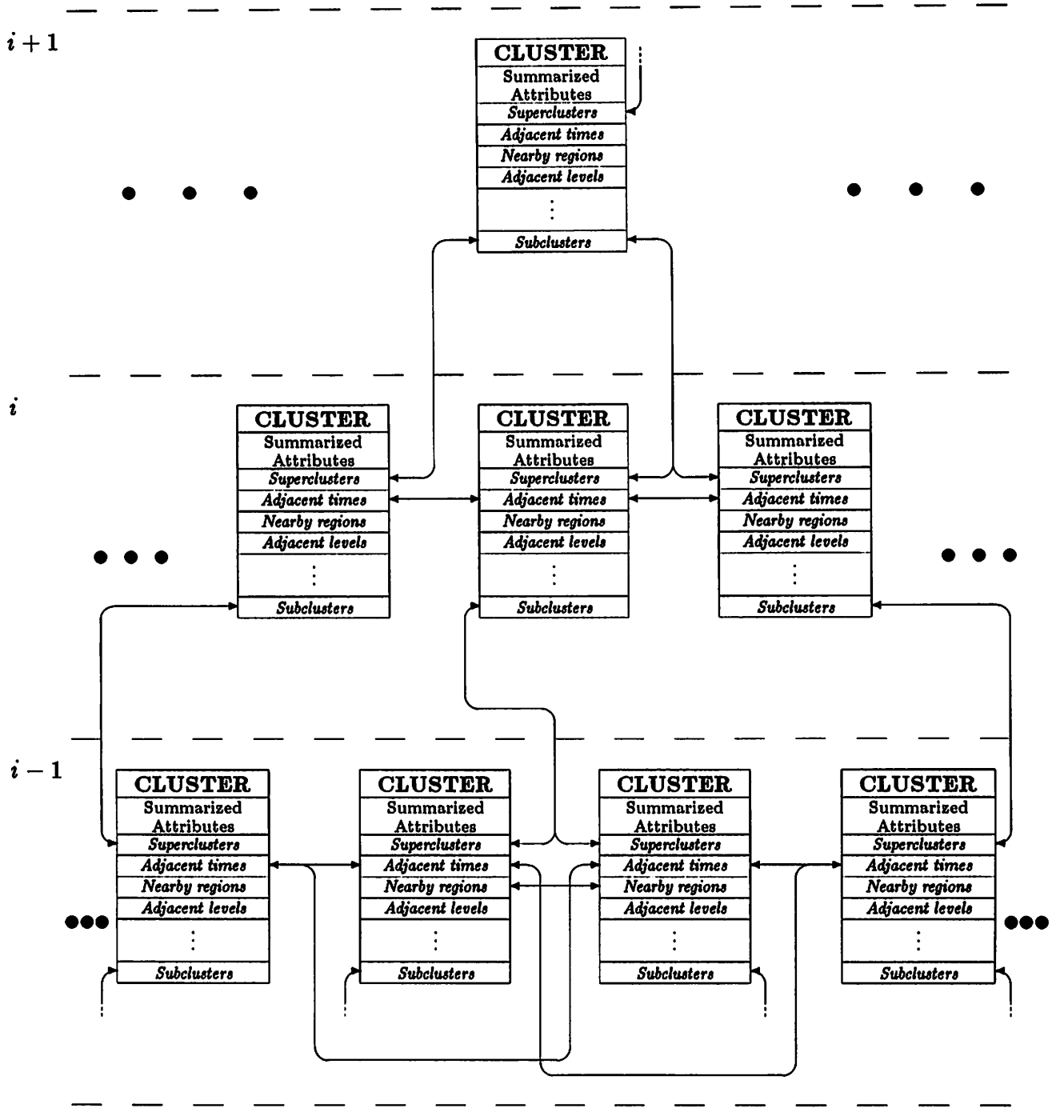
[15]Currently, bias is represented as a set of 8 numeric parameters corresponding to: depth-first-by-belief, breadth-first-by-belief, depth-first-by-level, breadth-first-by-level, node-independence, tracking-stress, time-investment, and goal-weighting. These parameters are used to guide the node's decisions about where in the solution space it will concentrate. Bias has been been studied in depth in learning research [59].

perception of the problem solving situation (including the goals of cooperation), and since effective cooperation requires that agents have some common views about the problem and each other (as discussed in Section 2.4), it is in the agents' self-interests to maximize the rewards of cooperation by maintaining compatible views of the situation through their biases. Since bias plays such a crucial role in problem solving and cooperation, techniques for detecting and responding to inappropriate bias will be an important aspect of the proposed mechanisms.

The new implementation allows *incremental clustering*: the hypotheses can be clustered based on one attribute, the resulting clusters can be further clustered based on another attribute, and so on. At each step, the new clustering attribute depends on the node's bias, and the result is the formation of a clustering hierarchy. The abstracted blackboard is thus represented by a clustering hierarchy where higher-level clusters abstract the information of one or more lower-level clusters. More or less detailed views of the problem solving situation can therefore be found by accessing the appropriate clustering level. In addition, clusters at the same level are linked by their relationships; for example, they might represent adjacent time frames, nearby regions, or adjacent blackboard levels. The abstracted blackboard is a network of frame-like clusters (Figure 10), and it is hoped that this structure (along with the mechanisms to create it and use it for planning) will be useful in other domains where large numbers of details (facts, hypotheses) make recognition of more general situations difficult. For example, in planning a legal argument, a lawyer must sift through the facts of a case to find important abstract issues which can then be used to find similar cases. By ignoring the less important relationships or *dimensions* of a case [2], a lawyer can form increasingly abstract views, and these views could be represented in the same type of structure as has been outlined here.

Incremental clustering ceases when an important situation is recognized. For example, at the highest clustering level one cluster might be significantly higher rated (based on its attributes) than any other cluster and therefore represents the portion of the problem space to be worked on. The planner then uses this situation as a starting point for developing a plan. Unlike the preliminary implementation where the planner attempts to form plans in a predefined order, the improved planner uses the bias to decide what plans it should attempt to form. In forming these plans, the planner can move through the clustering hierarchy to find specific information.

Effective plans must incorporate predictive information as well, and this will be an important aspect of the proposed research. In particular, a plan should not only specify its expected results (the goal(s) it will achieve) but also when those results will be produced. Making such time predictions is complicated by the fact that KSs can require different amounts of time to execute in different situations. Predictive information for a plan should also specify possible reasons why the plan might be abandoned: the possible events or actions of other nodes that could make the plan unattractive or make another plan more attractive. This information improves the node's local activity since the node need only monitor for specific actions and events that will trigger a need to replan. In addition, this information can be useful to other nodes in determining the probability that another node will complete its plan. Thus, the planner will integrate may of the features of previous



At each clustering level, there may be more clusters that are not shown. Clusters at level *i-1* are clustered by their nearby-region relationships to get the clusters at level *i*, which in turn are clustered by their adjacent-time relationships to get clusters at level *i+1*. There may be any number of levels, and the order of relationships by which clustering is done can vary depending on the node's bias.

Figure 10: Part of a Clustering Hierarchy

planning systems, and it is hoped that this planner will provide a framework for planners in other domains (such as route planning or investment planning) where uncertainty in the environment and timing of actions makes such a planner necessary.[16]

5.1.2 Planning, Communication, and Control

The planner must not base plans solely on its high-level view of its own activity (the abstracted blackboard), but also on its models of the past, present, and intended future actions of other nodes. Therefore, the proposed implementation must develop mechanisms that allow nodes to form and exchange appropriate meta-level messages and that enable nodes to locally store received meta-level information. In the preliminary implementation, the messages were formed by summarizing the data into a set of numbers, while the details of communication were sidestepped by allowing nodes to look at each other's meta-level information.

In the proposed implementation of the abstracted blackboard, each cluster has a set of attributes that summarize the data encompassed in the cluster. These attributes include the range of beliefs of the data, the blackboard levels and the times and regions covered by the data, the amount of data, and the number of pending KSIs stimulated by the data. As in the preliminary implementation, this summarization can be equally well understood by all nodes since it does not refer to any specific local data structures but rather summarizes their attributes. In essence, this information represents a very large hypothesis (covering several levels, times, regions, and beliefs).

Intelligent domain-level communication (exchange of hypotheses and goals) requires nodes to reason about the relevance, timeliness, and completeness of each message to make appropriate communication decisions [20]. Communication knowledge sources and policies for exchanging meta-level information must be based on the same considerations; meta-level information should be selectively communicated when a node believes that the resulting improvement in network performance warrants the added communication overhead. Furthermore, there are tradeoffs to be considered as to when nodes should communicate about each other's plans and when they should infer each other's plans based on past meta-level information. Meta-level communication KSs may therefore closely resemble domain-level mechanisms. For example, it may be useful for nodes to not only send and receive meta-level messages, but to send and receive messages that request certain meta-level information. Since these messages affect how nodes view each other, communication decisions can lead to interesting behavior. A node with certain preferences, for example, might send only a selected subset of its meta-level information to lead other nodes into particular cooperative interactions. The self-interested nodes might misrepresent themselves to some extent in order to alter network behavior to their benefit.

When a node receives a domain-level message, it inserts the hypothesis or goal onto

[16] Note that such a planner assumes that it can plan activities for the near future with only a vague notion of later activities. In domains where activities highly constrain each other (where a node could "paint itself into a corner"), a planner that uses constraints to form an entire detailed plan may be more suitable [56]. However, in unpredictable environments, an effective planner must be able to plan its current activities without depending on future states that may never arrive.

the appropriate blackboard. At the meta-level, this would correspond to placing information from other nodes into the abstracted blackboard or the plan queue. This may be inappropriate because a node does not utilize a received plan in the same way that it uses a received hypothesis or goal; it cannot treat a received plan as if it were created locally since it may depend on data that is not available locally. A more promising approach that will be used in this research is to store this information in separate but similar data structures. Each node would thus have local models of other nodes (peer-images [11], belief subsystems [36], process models [31]), that it could use to better plan cooperative interactions with those nodes.

The planner will therefore have a complicated task. It forms plans based on the available KSIs (from goal processing), on its high-level local view (the abstraction blackboard), and on its network awareness (the static organizational structure and the more dynamic local models of other nodes plans and actions). Since its plans and predictions are based on uncertain information (uncertainty about events, uncertainty about when planned actions will occur and of their results, uncertainty about the predictions of other nodes' actions, etc.), a node is not assured of making correct planning decisions.

Making intelligent planning decisions can be a costly activity [4,33,48]. The preliminary results indicated that expending the computation and communication resources to make informed decisions is worthwhile in complex problem solving situations. Studies of the costs as well as the benefits of the proposed mechanisms will continue to be part of the research. In addition, it is hoped that techniques might be developed that will allow the planner to recognize when the expected resource investment to improve a decision outweighs the expected improvement in that decision. In effect, just as the planner controls domain-level activity (the KS activity), a planner-controller should be developed to control the activities of the planner.

The planner as described can only choose from among the tasks currently resident on the node—it only controls the local activity of the node. To redistribute tasks among the nodes (for example, if some nodes were idle while others were overwhelmed with tasks), network control must be incorporated into the distributed problem solving framework. Network control can take the form of passing tasks among nodes, of reorganizing the network, or both [20]. As part of this proposed research, task passing mechanisms will be incorporated into the nodes. With its improved network awareness (from the planner and meta-level information), a node can make informed decisions about what tasks to send and where to send them. In essence, by exchanging the appropriate meta-level information, the nodes negotiate to make these network control decisions [47,54]. This research will therefore combine ideas from all three approaches to distributed problem solving (Section 3.1.1): to the underlying FA/C framework will be added negotiation for task passing and some elements of multi-agent planning so that each node can predict its interactions with other nodes.

Organizational self-design mechanisms are beyond the scope of this research. However, it is hoped that the proposed mechanisms will help further the eventual development of these mechanisms. Since these mechanisms will depend on nodes having network awareness to detect inappropriate organizations and to suggest better organizations, the pro-

posed research will develop a foundation for acquiring the requisite information to achieve organizational self-design.

5.1.3 Expected Results

Once implemented, the set of proposed mechanisms will be tested in a variety of simulated problem solving environments, including large networks (tens of nodes) and smaller networks that will be studied in detail. Characteristics of the environments, such as the assignment of expertise to nodes and the timing of the incorporation of sensor data into nodes will be varied to learn how the mechanisms help nodes to cooperate more effectively to achieve the goals of cooperation. Furthermore, it is likely that the mechanisms will improve cooperation more in some environments than in others, and this will lead to an empirical evaluation of the strengths and weaknesses of the mechanisms. Determining the efficacy of these mechanisms will require analysis of the environments to find benchmark values of optimal and reasonable performance to use for comparison [20], and new analytical techniques may need to be developed.

The mechanisms will also add overhead to problem solving in each node. In the preliminary research, the overhead was justified in complex problem solving situations where planning and meta-level communication allowed nodes to avoid large amounts of unnecessary or redundant processing. Due to their sophistication, the proposed mechanisms may be very costly to a node. Not only should an empirical evaluation of the costs and benefits of these mechanisms be performed, but techniques for allowing a node to determine the extent to which it need employ these mechanisms and to adapt its activities accordingly should also be explored. Specifically, a node should decide whether the benefits of improving its model of another node justify the costs, and must judge whether it would be more expedient to update this model through communication or through making predictions based on its current model.

It is hoped that these mechanisms will not only increase node self- and network-awareness, but that they might also aid observers of the network (human experimenters, network monitoring systems) to understand node problem solving decisions and network problem solving activity. Previous research has been devoted to developing an abstract view of network problem solving activity so that important actions and interactions can be recognized [46]. The proposed mechanisms can aid this process by providing high-level views of node intentions and predicted interactions: large, complex problem solving networks can be understood in terms of plans and clusters rather than more detailed KSIs and hypotheses. Also, the detection and diagnosis of faults in the network [35] might similarly benefit from these mechanisms since symptoms of faults may be spread out in sequences of actions and may only be recognizable with a high-level view.

Finally, the empirical studies of diverse distributed problem solving environments that will be performed to evaluate these mechanisms may also lead to new insights into cooperation among problem solvers. The set of environments that have been studied to date has been limited by the complexity of analysis for larger and more complicated networks as well as by the limitations of the available computational resources. It is hoped that the new mechanisms will allow this set to be extended so that new cooperation phenomena

can be observed and analyzed.

5.2 Applications in Other Domains

Most of the proposed research concerns developing mechanisms to improve cooperation in the problem solving network simulated by the DVMT. It is hoped, however, that the basic approach embodied in these mechanisms, that of integrating planning, predicting, meta-level communication, and network control, will prove appropriate in other distributed computing applications as well.

In distributed problem solving, this approach can achieve all the benefits of multi-agent planning and of negotiation in environments where these are useful techniques. For example, when uncertainty about the effects and duration of agent actions is reduced, and when communication between agents is rapid compared to action durations, then agents might iteratively form and exchange plans until they converge on an appropriate multi-agent plan—all before they begin to act. This technique for forming multi-agent plans was described by Rosenschein and Genesereth [49]. As outlined above, the mechanisms proposed for exchanging tasks among nodes can achieve the benefits of negotiation. Furthermore, in distributed problem solving networks based on negotiation, incoherence occurs because nodes cannot predict what tasks may become available in the future; the proposed mechanisms might allow nodes to make predictions about these tasks and therefore increase coherence in this type of distributed problem solving network.

It is hoped that the mechanisms as well as the approach developed in this research can be applied in other domains. The mechanisms for forming and representing the abstract view of the problem could be reimplemented in any domain where relationships between details (facts, hypotheses) can be defined and where reasons for preferring some relationships to others can be represented. The planner will be implemented to guide a blackboard-based problem solver, but many of the algorithms, heuristics, and techniques it employs that use available knowledge about local and network activity when making decisions should be useful in other problem solving architectures as well. It is also hoped that the techniques and heuristics for making these decisions can be generalized to other problem solving domains where a planner may only have a vague view of the future environment. Finally, the experiences gained in developing mechanisms to intelligently generate, exchange, and use meta-level information will hopefully have utility in distributed computing systems where computing elements need models of network activity for coordination.

Most distributed computing systems have assumed that the tasks that each computing agent executes are independent of the tasks at other agents. As distributed computation is used in more diverse applications, however, the task independence assumption becomes invalid. For example, in a distributed programming environment, each agent may have pieces of several distributed programs. Because the pieces of a distributed program may need to synchronize or exchange information, each agent must consider how the timing of its local decisions will affect activity on other agents. To achieve smooth cooperation between these agents, the agents will need predictions about each other's future decisions. Although the specific mechanisms needed by such a system may differ from those implemented in the

DVMT, the fundamental approach of using local planning and meta-level communication to improve cooperation appears to be generally applicable to all distributed computing systems where agents have interacting tasks. In fact, this approach was used in developing a distributed version of the DVMT [19].

5.3 Anticipated Contributions

The proposed research is expected to contribute to both theoretical and practical issues in cooperation among problem solvers. From a theoretical viewpoint, the research will be concerned with what types of knowledge and interactions are needed for artificial systems to cooperate coherently. Cooperation among natural systems (such as people) appears to be based on inherent common knowledge (generalizations about human behavior) augmented with communicated situation-specific knowledge. It is hoped that creating artificial systems that cooperate coherently to achieve a variety of goals will improve our understanding of cooperation among natural systems.

In more practical terms, the research should result in the development of mechanisms that improve distributed problem solving performance and that integrate local and network control. The strengths and limitations of these mechanisms will be extensively evaluated and will hopefully lead to a general, qualitative summary of what mechanisms are appropriate in what situations. Finally, the mechanisms and the basic approach that they represent will be evaluated both in terms of where they might lead in the DVMT (towards an organizational self-design component, for example) and in terms of how they might be used in other distributed computing applications.

5.4 Summary

In environments where cooperative behavior is beneficial, it is important that agents exist that are capable of achieving these benefits. An intelligent agent can respond to a situation in diverse ways and must apply its knowledge to decide upon the best response. Since the suitability of a response depends on how that response will affect the future, the agent must be able to predict future situations (and responses). That is, an intelligent agent must be able to plan worthwhile sequences of actions. Furthermore, if the agent shares its environment with other agents, it must take their current and planned future actions into account when it forms its own plans.

This paper outlines research to study mechanisms for achieving effective cooperation between intelligent problem solving agents that work together to solve a single problem. The proposed approach concentrates on the strong connection between distributed planning and distributed problem solving: effective distributed problem solving requires that agents plan their actions and interactions rather than making isolated and short-sighted decisions. To plan their actions and interactions, agents must have sophisticated planners that can: interleave planning and execution, monitor and revise inappropriate plans, make temporal predictions about when actions and events may occur, and use models of other agents to develop plans that will allow the agent to interact favorably with other agents. This last requirement necessitates the development of mechanisms that allow agents to

exchange information so that each maintains an adequate model of the others. In addition, since the form cooperation takes depends on the characteristics of the problem and of the problem solving network, the mechanisms that are developed must be applicable in a range of situations.

Much of this research will integrate and extend research cited in the literature. The planner will integrate ideas from several sources (see Section 3.2.4). The meta-level communication mechanisms will be based on those used to communicate hypotheses and goals in the DVMT [14,22]. The network control mechanisms will use bidding protocols as their foundation [47,54,55]. Finally, empirically evaluating the new mechanisms will be possible thanks to the work of Corkill, Lesser, and numerous graduate students who have constructed the DVMT.

In summary, the principal issues to be addressed in the proposed research include:

- Developing a flexible high-level representation of an agent's problem state to permit the recognition of a variety of problem solving situations.
- Developing a planner that can use this representation to generate plans to achieve important high-level goals.
- Providing the planner with the ability to make predictions: temporal predictions about when important results will be derived, and predictions about possible causes for plan abandonment that can be used to monitor plan execution.
- Providing the planner with models of other agents' past, present, and intended future activities so that important interactions can be predicted and planned around.
- Developing communication mechanisms that enable agents to intelligently use the communication resources to update their models of each other.
- Developing network control mechanisms that allow agents to cooperate by exchanging tasks and responsibilities.
- Enumerating a set of important goals of cooperation and applying the mechanisms in a variety of cooperative distributed problem solving situations that embody these goals.
- Studying the benefits, costs, and limitations of these mechanisms through experimentation.
- Generalizing the approach to understand the types of knowledge and interactions needed for coherent cooperation and outlining possible applications for this approach in other distributed systems.

Acknowledgments

I would like to thank the members of my dissertation committee for their invaluable assistance as I developed this dissertation proposal. My advisor, Victor Lesser, has introduced me to many fascinating issues in distributed problem solving and has both provided valuable insights and asked challenging questions as I prepared this proposal. Dan Corkill's pragmatism has forced me to recognize many important issues in implementing my proposed mechanisms, and his suggestions have helped me to avoid making broad statements that I may have regretted. Krithi Ramamritham helped me to clarify this proposal by pointing out the use of AI jargon and assumptions, and his interest in achieving cooperation in other distributed systems has continually prompted me to consider the more general applicability of the proposed mechanisms. Reid Smith has helped me to recognize many of the implicit assumptions I was making about distributed problem solving networks and domains, and with his help I have made these assumptions explicit and have therefore better defined the range of applicability of the approach.

I would also like to thank Jim Kurose for introducing me to economic models of planning and cooperation, and Les Gasser for several suggestions concerning both the content and presentation of this proposal.

REFERENCES

[1] James F. Allen. "Towards a general theory of action and time." *Artificial Intelligence*, **23**(1984):123–154.

[2] Kevin D. Ashley. "Reasoning by analogy: a survey of selected AI research with implications for legal expert systems." *Proceedings of the First Annual Conference on Law and Technology*, Houston, Texas, 1984.

[3] Robert Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1984.

[4] Jeffrey A. Barnett. "How much is control knowledge worth?: A primitive example." *Artificial Intelligence*, **22**(1984):77–89.

[5] Andrew G. Barto. "Learning by statistical cooperation of self-interested neuron-like computing elements." Technical Report 85-11, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, April 1985.

[6] Bertram Bruce and Denis Newman. "Interacting Plans." *Cognitive Science*, **2**(1978):195–233.

[7] Stephanie Cammarata, David McArthur, and Randall Steeb. "Strategies of cooperation in distributed problem solving." *Proceedings of the Eighth International Conference on Artificial Intelligence*, pages 767–770, August 1983.

[8] Peter Cheeseman. "A representation of time for automatic planning." *Proceedings of the IEEE International Conference on Robotics*, pages 513–518, 1984.

[9] R. T. Chien and S. Weissman. "Planning and execution in incompletely specified environments." *Proceedings of the Fourth International Conference on Artificial Intelligence*, pages 169–174, 1975.

[10] Daniel D. Corkill. "Hierarchical planning in a distributed environment." *Proceedings of the Sixth International Conference on Artificial Intelligence*, pages 168–175, August 1979.

[11] Daniel D. Corkill. "An organizational approach to planning in distributed problem solving systems." Technical Report 80-13, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, May 1980.

[12] Daniel D. Corkill and Victor R. Lesser. "A goal-directed Hearsay-II architecture: Unifying data and goal directed control." Technical Report 81-15, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, June 1981.

[13] Daniel D. Corkill, Victor R. Lesser, and Eva Hudlicka. "Unifying data-directed and goal-directed control: An example and experiments." *Proceedings of the Second National Conference on Artificial Intelligence*, pages 143–147, August 1982.

[14] Daniel David Corkill. *A Framework for Organizational Self-Design in Distributed Problem Solving Networks.* Ph.D. Dissertation, University of Massachusetts, February 1983. (Available as Technical Report 82-33, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1982.)

[15] Daniel D. Corkill and Victor R. Lesser. "The use of meta-level control for coordination in a distributed problem solving network." *Proceedings of the Eighth International Conference on Artificial Intelligence*, pages 748–756, August 1983.

[16] Randall Davis. "A model for planning in a multi-agent environment: Steps toward principles of teamwork." MIT AI Working Paper 217, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, Massachusetts, June 1981.

[17] Randall Davis and Reid G. Smith. "Negotiation as a metaphor for distributed problem solving." *Artificial Intelligence*, **20**(1983):63–109.

[18] Richard Dawkins. *The Selfish Gene.* Oxford University Press, New York, 1976.

[19] Edmund H. Durfee, Daniel D. Corkill, and Victor R. Lesser. "Distributing a Distributed Problem Solving Network Simulator." *Proceedings of the Fifth Real-time Systems Symposium*, pages 237–246, December 1984.

[20] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. "Coherent Cooperation Among Communicating Problem Solvers." Technical Report 85-15, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, April 1985.

[21] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. "Increasing coherence in a distributed problem solving network." *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 1025–1030, August 1985.

[22] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. "Making coherent communication decisions in a distributed problem solving network." *Proceedings of the Workshop on AI and Distributed Problem Solving, National Academy of Sciences*, pages 59–75, May 1985.

[23] Manfred Eigen and Ruthild Winkler. *Laws of the Game.* Harper and Row, New York, 1981.

[24] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, D. Raj Reddy. "The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty." *Computing Surveys*, **12**(2):213–253, June 1980.

[25] Michael Fehling and Lee Erman. "Report on the Third Annual Workshop on Distributed Artificial Intelligence." *SIGART Newsletter*, **84**:3–12, April 1983.

[26] Jerome A. Feldman and Robert F. Sproull. "Decision theory and artificial intelligence II: The hungry monkey." *Cognitive Science*, **1**(1977):158–192.

[27] R. E. Fikes and N. J. Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving." *Artificial Intelligence*, **2**(1971):189–208, 1971.

[28] R. E. Fikes. "A commitment-based framework for describing informal cooperative work." *Cognitive Science*, **6**(1982):331–347, 1982.

[29] Mark S. Fox. "Constraint-directed search: A case study of job-shop scheduling." Ph.D. Dissertation, Carnegie-Mellon University, December 1983. (Available as Technical Report CMU-CS-83-161, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213, December 1983.)

[30] Michael Georgeff. "Communication and interaction in multi-agent planning." *Proceedings of the Eighth International Conference on Artificial Intelligence*, pages 125–129, August 1983.

[31] Michael Georgeff. "A theory of action for multiagent planning." *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 121–125, August 1984.

[32] Frederick Hayes-Roth and Victor R. Lesser. "Focus of attention in the Hearsay-II speech understanding system." *Proceedings of the Fifth International Conference on Artificial Intelligence*, pages 27–35, August 1977.

[33] Barbara Hayes-Roth. "A blackboard model of control." *Stanford Heuristic Programming Project Report No. HPP-83-38*, revised August 1984.

[34] G. M. Heal. *The Theory of Economic Planning.* American Elsevier, New York, 1973.

[35] Eva Hudlicka and Victor Lesser. "Meta-level control through fault detection and diagnosis." *Proceedings of The Fourth National Conference on Artificial Intelligence*, pages 153–161, August 1984.

[36] Kurt Konolige. "A deductive model of belief." *Proceedings of the Eighth International Conference on Artificial Intelligence*, pages 377–381, August 1983.

[37] J. F. Kurose, M. Schwartz, and Y. Yemini. "A microeconomic approach to optimization of channel access policies in multiaccess networks." *Proceedings of the Fifth International Symposium on Distributed Computing Systems*, pages 70–80, May 1985.

[38] Victor R. Lesser and Lee D. Erman. "An experiment in distributed interpretation." *IEEE Transactions on Computers*, **C-29**(12):1144–1163, December 1980.

[39] Victor R. Lesser and Daniel D. Corkill. "Functionally accurate, cooperative distributed systems." *IEEE Transactions on Man, Systems, and Cybernetics*, **SMC-11**(1):81–96, January 1981.

[40] Victor Lesser, Daniel Corkill, Jasmina Pavlin, Larry Lefkowitz, Eva Hudlicka, Richard Brooks, and Scott Reed. "A high-level simulation testbed for cooperative distributed problem solving." *Proceedings of the Third International Conference on Distributed Computer Systems*, pages 341–349, October 1982.

[41] Victor R. Lesser and Daniel D. Corkill. "The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem solving networks." *AI Magazine*, 4(3):15–33, Fall 1983.

[42] John D. Lowrance and Daniel P. Friedman. "Hendrix's model for simultaneous actions and continuous processes: An introduction and implementation." *International Journal of Man-Machine Studies*, 9(1977):537–581.

[43] Gordon I. McCalla, Larry Reid, and Peter F. Schneider. "Plan creation, plan execution, and knowledge acquisition in a dynamic microworld." *International Journal of Man-Machine Studies*, **16**(1982):89–112.

[44] Drew McDermott. "A temporal logic for reasoning about processes and plans." *Cognitive Science*, **6**(1982):101–155.

[45] Horst Oberquelle, Ingbert Kupka, and Susanne Maass. "A view of human-machine communication and co-operation." *International Journal of Man-Machine Studies*, **19**(1983):309–333.

[46] Jasmina Pavlin and Daniel D. Corkill. "Selective abstraction of AI system activity." *Proceedings of The Fourth National Conference on Artificial Intelligence*, pages 264–268, August 1984.

[47] Krithivasan Ramamritham and John A. Stankovic. "Dynamic task scheduling in hard real-time distributed systems." *IEEE Software*, pages 65–75, July 1984.

[48] Jeffrey S. Rosenschein and Vineet Singh. "The utility of meta-level effort." *Stanford Heuristic Programming Project Report No. HPP-83-20*, March 1983.

[49] Jeffrey S. Rosenschein and Michael R. Genesereth. "Communication and cooperation." *Stanford Heuristic Programming Project Report No. HPP-84-5*, October 1984.

[50] Jeffrey S. Rosenschein and Michael R. Genesereth. "Deals among rational agents." *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 91–99, August 1985. 1984.

[51] Earl D. Sacerdoti. *A Structure for Plans and Behavior.* Elsevier North-Holland, New York, 1977.

[52] Earl D. Sacerdoti. "Problem solving tactics." *Proceedings of the Sixth International Conference on Artificial Intelligence*, pages 1077–1085, 1979.

[53] Martin Shubik. *Game Theory in the Social Sciences: Concepts and solutions.* The MIT Press, Massachusetts, 1982.

[54] Reid G. Smith. "The contract-net protocol: High-level communication and control in a distributed problem solver." *IEEE Transactions on Computers*, **C-29**(12):1104–1113, December 1980.

[55] Reid G. Smith and Randall Davis. "Frameworks for cooperation in distributed problem solving." *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-11**(1):61–70, January 1981.

[56] Mark Stefik. "Planning with Constraints." *Artificial Intelligence*, **16**(1981):111–140, 1981.

[57] Austin Tate. "Interacting goals and their use." *Proceedings of the Fourth International Conference on Artificial Intelligence*, pages 215–218, 1975.

[58] Austin Tate. "Generating project networks." *Proceedings of the Fifth International Conference on Artificial Intelligence*, pages 888–893, 1977.

[59] Paul E. Utgoff. "Shift of bias for inductive concept learning." *Machine Learning, Volume 2*, Morgan Kaufman, 1986.

[60] Steven A. Vere. "Planning in time: Windows and durations for activities and goals." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-5**(3):246–67, May 1983.

[61] Richard Waldinger. "Achieving several goals simultaneously." *Machine Intelligence 8*, pages 94–136, 1977.

[62] Robert B. Wesson. "Planning in the world of the air traffic controller." *Proceedings of the Fifth International Conference on Artificial Intelligence*, pages 473–479, 1977.

[63] David E. Wilkins. "Domain-independent planning: Representation and plan generation." *Artificial Intelligence*, **22**(1984):269–301.